```sql
-- 🎯 PROJECT: Online Bookstore SQL Data Analysis

-- 💻 Database: PostgreSQL

-- 📁 Includes: Table creation, CSV import, and SQL queries (basic to advanced)

-- -------------------------------




-- 🔶 Step 1: Create the Database

CREATE DATABASE OnlineBookstore;




-- 🔶 Step 2: Switch to the Database

\c OnlineBookstore;




-- 🔶 Step 3: Create Tables

DROP TABLE IF EXISTS Books;

CREATE TABLE Books (

   Book_ID SERIAL PRIMARY KEY,

   Title VARCHAR(100),

   Author VARCHAR(100),

   Genre VARCHAR(50),

   Published_Year INT,

   Price NUMERIC(10, 2),

   Stock INT

);
```

```
DROP TABLE IF EXISTS customers;
CREATE TABLE Customers (
    Customer_ID SERIAL PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100),
    Phone VARCHAR(15),
    City VARCHAR(50),
    Country VARCHAR(150)
);


DROP TABLE IF EXISTS orders;
CREATE TABLE Orders (
    Order_ID SERIAL PRIMARY KEY,
    Customer_ID INT REFERENCES Customers(Customer_ID),
    Book_ID INT REFERENCES Books(Book_ID),
    Order_Date DATE,
    Quantity INT,
    Total_Amount NUMERIC(10, 2)
);


-- 🔶 Step 4: Preview Tables (Optional)
SELECT * FROM Books;
SELECT * FROM Customers;
SELECT * FROM Orders;
```

```sql
-- 🔶 Step 5: Import CSV Data into Tables
COPY Books(Book_ID, Title, Author, Genre, Published_Year, Price, Stock)
FROM 'C:\SQL Projects\Bookstore_by_Satish_Dhawale\Books.csv'
CSV HEADER;

COPY Customers(Customer_ID, Name, Email, Phone, City, Country)
FROM 'C:\SQL Projects\Bookstore_by_Satish_Dhawale\Customers.csv'
CSV HEADER;

COPY Orders(Order_ID, Customer_ID, Book_ID, Order_Date, Quantity, Total_Amount)
FROM 'C:\SQL Projects\Bookstore_by_Satish_Dhawale\Orders.csv'
CSV HEADER;

-- ============================================
-- 🟦 BASIC SQL QUERIES
-- ============================================

-- 1) Retrieve all books in the "Fiction" genre
SELECT * FROM Books
WHERE Genre='Fiction';

-- 2) Find books published after the year 1950
SELECT * FROM Books
WHERE Published_year > 1950;

-- 3) List all customers from Canada
SELECT * FROM Customers
WHERE Country='Canada';
```

```sql
-- 4) Show orders placed in November 2023

SELECT * FROM Orders

WHERE Order_Date BETWEEN '2023-11-01' AND '2023-11-30';


-- 5) Retrieve the total stock of books available

SELECT SUM(Stock) AS Total_Stock

FROM Books;


-- 6) Find the details of the most expensive book

SELECT * FROM Books

ORDER BY Price DESC

LIMIT 1;


-- 7) Show all customers who ordered more than 1 quantity of a book

SELECT * FROM Orders

WHERE Quantity > 1;


-- 8) Retrieve all orders where the total amount exceeds $20

SELECT * FROM Orders

WHERE Total_Amount > 20;


-- 9) List all genres available in the Books table

SELECT DISTINCT Genre FROM Books;


-- 10) Find the book with the lowest stock

SELECT * FROM Books

ORDER BY Stock

LIMIT 1;
```

```sql
-- 11) Calculate the total revenue generated from all orders

SELECT SUM(Total_Amount) AS Revenue

FROM Orders;


-- ==============================================

-- 🟩 ADVANCED SQL QUERIES

-- ==============================================


-- 1) Retrieve the total number of books sold for each genre

SELECT b.Genre AS Genre, SUM(o.Quantity) AS Total_Books_Sold

FROM Orders o

JOIN Books b ON o.Book_ID = b.Book_ID

GROUP BY b.Genre;


-- 2) Find the average price of books in the "Fantasy" genre

SELECT ROUND(AVG(Price), 2) AS Average_Price

FROM Books

WHERE Genre = 'Fantasy';


-- 3) List customers who have placed at least 2 orders

SELECT o.Customer_ID, c.Name, COUNT(o.Order_ID) AS ORDER_COUNT

FROM Orders o

JOIN Customers c ON o.Customer_ID = c.Customer_ID

GROUP BY o.Customer_ID, c.Name

HAVING COUNT(o.Order_ID) >= 2;
```

```
-- 4) Find the most frequently ordered book

SELECT o.Book_ID, b.Title, COUNT(o.Order_ID) AS ORDER_COUNT

FROM Orders o

JOIN Books b ON b.Book_ID = o.Book_ID

GROUP BY o.Book_ID, b.Title

ORDER BY ORDER_COUNT DESC

LIMIT 1;



-- 5) Show the top 3 most expensive books of 'Fantasy' Genre

SELECT * FROM Books

WHERE Genre = 'Fantasy'

ORDER BY Price DESC

LIMIT 3;



-- 6) Retrieve the total quantity of books sold by each author

SELECT b.Author, SUM(o.Quantity) AS TOTAL_BOOKS_SOLD

FROM Orders o

JOIN Books b ON o.Book_ID = b.Book_ID

GROUP BY b.Author;



-- 7) List the cities where customers who spent over $30 are located

SELECT DISTINCT c.City, o.Total_Amount

FROM Orders o

JOIN Customers c ON c.Customer_ID = o.Customer_ID

WHERE Total_Amount > 30;
```

```sql
-- 8) Find the customer who spent the most on orders

SELECT c.Customer_ID, c.Name, SUM(o.Total_Amount) AS TOTAL_SPENT

FROM Orders o

JOIN Customers c ON c.Customer_ID = o.Customer_ID

GROUP BY c.Customer_ID, c.Name

ORDER BY TOTAL_SPENT DESC

LIMIT 1;


-- 9) Calculate the stock remaining after fulfilling all orders

SELECT b.Book_ID, b.Title, b.Stock,

COALESCE(SUM(o.Quantity), 0) AS Order_Quantity,

b.Stock - COALESCE(SUM(o.Quantity), 0) AS Remaining_Quantity

FROM Books b

LEFT JOIN Orders o ON b.Book_ID = o.Book_ID

GROUP BY b.Book_ID

ORDER BY b.Book_ID;
```